# WHAT MAKES FOR A HIT POP SONG? WHAT MAKES FOR A POP SONG?

NICHOLAS BORG AND GEORGE HOKKANEN

Abstract. The possibility of a hit song prediction algorithm is both academically interesting and industry motivated. While several companies currently attest to their ability to make such predictions, publicly available research suggests that current methods are unlikely to produce accurate predictions. Support Vector Machines were trained on song features and YouTube view counts to very limited success. We discuss the possibility that musical features alone cannot account for popularity. Given the lack of substantial findings in popularity position, we attempted a more feasible project. Current research into automated genre detection given features extracted from music has shown more promising. Using a combination of K-Means clustering and Support Vector Machines, as well as a Random Forest, we produced two automated classifiers that performs five times better than chance for 10 genres.

## 1. Introduction

## 2. Data

Our main source of data was the Million Song Dataset Subset, distributed by Labrosa. The subset provides pre-extracted features for 10000 songs, including: song and artist names, song duration, timbral data (MFCC-like features) and spectral data for the entire song, number of sections and average section length, and a number of other features.
In order to measure the popularity of a song, for each song we collected the number of view counts registered on the video returned as the first link in a YouTube search using the YouTube API, the query being the song's and artist's names. We checked by hand the accuracy of this scraping method and concluded that the two errors in thirty randomly drawn songs was not a problem given that the errors also coordinate well with very low view counts (i.e. something unpopular enough to not return a copy of the song on youtube ends up returning an unrelated video with a low view count). For Genre Classification, we used the Million Song Dataset Genre subset. The dataset includes features extracted from 59,600 songs divided into ten genres: classic pop and rock, folk, dance and electronica, jazz and blues, soul and reggae, punk, metal, classical, pop, and hip-hop. Features for each song include loudness, tempo, time signature, key, mode, duration, as well as average timbral data and average timbral variance.

## 3. Preliminary Analysis: Popularity Prediction

We first ran basic correlation coefficients between different parts of the metadata and also with our extracted youtube view counts. The results were largely insignificant and included weak correlations such as one of .2 between the tempo and loudness metadata features. Correlations between the youtube view counts and the echonest metadata features loudness, tempo, hotttness, and danceability were completely negligible (less than .05 in magnitude). The fact that these are not at all correlated is interesting in its own right because it points to no single metadata feature being at all a good predictor of views on youtube.

## 4. Popularity Prediction – Methodology and Results

4.1. **Linear Classification using Support Vector Machines.** First we note that the feature vectors given by the million song dataset are not of uniform length as the spectral and mfcc coefficients are provided for intervals of the song and thus the number of those features depends on the length of the song. To account for all of the data and compress it to uniform length for each song, we took averages of the coefficients for each half, fourth, or sixth of the song and concatenated the result for feature vectors of length 24, 48, or 72 for each set of coefficients, and then normalized the resulting lists of feature vectors. We trained SVMs with each extracted feature thinking that some number of segments would outperform others (perhaps as they become closer to the average of the actual number of segments in the songs). We altered the cost, bias, and kernel, but the precision never gained more than one percent on our bias. That is, 53% of the songs had youtube view counts over 10K and 19% were over 100K. The SVMs regardless of feature choice and parameters never achieved more than 53% and 81.5% precision. The recall was always less than 53.5% and 82%.

Finally, we tried an SVM on spectral averages and metadata features including tempo, time signature, energy, loudness, duration, the number of sections, and finally the Echonest's popularity measure, 'hotttnesss'. Trained on popularity measure of 100K views, these features resulted in accuracy no better than before. However, when trained on the popularity measure of 10K views, the precision rose from under 53% to over 55%. Using the same features without 'hotttnesss' results in the same performance as above (no better than the bias of the dataset). From this we conclude that the Echonest's 'hotttnesss' measure (for which they have not released an explanation as to how it is quantified) gives a very small amount of predictive power (2-3% above the bias) on whether or not a song will have more than 10K views on Youtube.

4.2. **String Kernel.** Given that the mfcc and spectral data is temporal, we wanted to use the ordering therein to describe the sound. Motivating a string kernel SVM approach, we create 'string' features for our songs as follows. For each i, we take the spectral bucket i (corresponding to a frequency-aggregate magnitude) for each spectra vector within a range of each song (usually about 45 seconds in the middle). This gives a list of values which correspond to the magnitudes over time of this slice of the sound spectrum for every song. Next, using a subset of this data (we used two hundred of the ten thousand songs) we compute a list of intervals (we used 26 of them, corresponding to the characters a through z) that uniformly distribute the data. Then using these intervals, we compute a string for each sequence of data obtained in the first step by replacing each value with a symbol or letter that represents the interval.

When we tried a string kernel SVM on this data, it was unable to converge at even the first iteration. Notably, string kernels are not guaranteed to be general Mercer kernels, and our string data was so long and varied that we suppose edit distance may be a very poor metric. Furthermore, edit distance is not aware of operations such as translation, which can help tell that two pieces of music are similar (consider putting a silent delay at the beginning of a song, then the method mentioned above will not recognize the songs as anything similar). For this reason, we conclude that this method of using a string kernel is a dead end and may only be helped by more general pattern matching metrics instead of the overly simplistic edit distance. Algorithms such as these, however, are their own area of research and fall under the categorization of structural segmentation and similarity metrics (most often used for

identifying song covers). We find the possibility of research on estimating popularity by structural segmentation to be interesting.

## 5. Popularity Prediction Discussion

## 6. Genre Classification

In light of our modest results at predicting popularity, we began to investigate another problem involving only musical features. (INCLUDE STUFF ABOUT RESEARCH HERE).

## 7. Genre Classification Results